UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/697,421	08/23/96	MOVALLI M	06555.0001-0

MM91/0721

FINNEGAN HENDERSON FARABOW GARRETTAND
DUNNER LLP
1300 I STREET NW
WASHINGTON DC 20005

EXAMINER	
TREMBLAY, M	
ART UNIT	**PAPER NUMBER**
2876	

DATE MAILED: 07/21/00

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
ASSISTANT SECRETARY AND COMMISSIONER OF
PATENTS AND TRADEMARKS
Washington, D.C. 20231

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Paper No. 24

Application Number: 08/697,421
Filing Date: 08/23/1996
Appellant(s): Movalli et al.

Jeffrey A. Berkowitz
For Appellant

EXAMINER'S ANSWER

This is in response to appellant's brief on appeal filed April 19, 2000.

(1) Real Party in Interest

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

(3) Status of Claims

The statement of the status of the claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Invention

The summary of invention contained in the brief is correct.

(6) Issues

The appellant's statement of the issues in the brief is correct.

(7) Grouping of Claims

Appellant's brief includes a statement that claims 1, 3-7, 9-23, and 25-31 do not stand or fall together and provides reasons as set forth in 37 CFR 1.192(c)(7) and (c)(8).

(8) Claims Appealed

The copy of the appealed claims contained in the Appendix to the brief is correct.

(9) Prior Art of Record

The following is a listing of the prior art of record relied upon in the rejection of claims under appeal.

Davies, Donald W., "Use of the 'Signature Token' to Create a Negotiable Document" from Charm, David "Advances in Cryptology: Proceedings of Crypto '83", pages 377-382.

4,825,050	Griffith et al.	4-1989
5,689,565	Spies et al.	11-1997

(10) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

Claims 1-4, drawn to methods for generating transactions, and claims 25-27, drawn to a system for generating secure endorsed transactions, are rejected under 35 U.S.C. § 103 as being unpatentable over Donald W. Davies "Use of the 'Signature Token' to Create a Negotiable Document ("Davies" hereinafter) in view of "in view of U.S. Patent #4,825,050 to Griffith et al. ("Griffith" hereinafter) . Davies discloses a computer implemented (page 378, under the heading "The Signature Token", e.g. the description of a smart card which has a "microprocessor") method of generating secure endorsed transactions (see description on page 378 under the

heading "The Signature Token", e.g. the description of debit cards, credit transactions, etc.), the method comprising:

receiving transaction data (9, 10, 11, 12, 14, and 15) corresponding to a transaction and at least one unique identifier of a customer (typically a human) (5); and

generating a unique code 16 from the transaction data and the unique identifier of a customer, wherein the unique code constitutes a secure endorsement of the transaction by the party corresponding to the unique identifier (5).

Davies discloses the features of the invention as described above, but does not teach that a "human" identifier, e.g. a biometric, can be used with such an encryption scheme to further enhance security. Griffith teaches that " Multiple inputs are accepted in the following manner: The individual information record 101 which is the data to be 'locked'; the individual identifier 100 which may be some characteristic of the individual e.g. finger, voice, or retinal pattern, signature, or chemical structure or some information known only to the individual, e.g. a combination, pass word or phrase; a private key 110 which is known only to the issuing entity and which is generated by any method 109 meeting the criteria for public key crypto systems outlined by W. Diffie and M.E. Hellman in their article cited above such as the system publicly disclosed by Rivest, Shamir, and Adleman ob cit; and optionally other data 113 which is necessary or convenient to include regarding the application made of the present method." Thus, Griffith teaches that the public/private key method can be used with a "human identifier" to thwart fraud. It would have been obvious at the time the invention was made to a person having ordinary skill in the art to include a "human identifier" as taught by Griffith in the system taught by Davies because this would make it more difficult for a thief to use a stolen smart card, as taught by Griffith to create the negotiable documents taught by Davies.

Re claim 25, Official Notice is taken that a card insertion sensor is notoriously old and well known in the art. It would have been obvious at the time the invention was made to a person having ordinary skill in the art to use a card insertion sensor in a smart card system according to

Davies and Even so that when a card is inserted into the device, the transaction can begin automatically, making the user's task easier, and speeding the transaction.

Claims 5-23 and 29-31 are rejected under 35 U.S.C. § 103 as being unpatentable over Davies in view of U.S. Patent #4,825,050 to Griffith et al. ("Griffith" hereinafter) and further in view of U.S. Patent #5,689,565 to Spies ("Spies" hereinafter). Davies and Griffith do not explicitly teach a network implementation or a need for a receipt, as mentioned above. A network implementation for a system like this is common. Spies provides an example of a networked system used for an application similar to the combined teachings of Davies and Griffith. It would have been obvious at the time the invention was made to a person having ordinary skill in the art to use a method according to the combined teachings of Davies and Griffith in a networked environment taught by Spies, because networked environments allow for the greater flexibility in performing transactions, as was commonly and well understood in the art.

(11) Response to Argument

Overview

Claim 1 sets the broad outlines of this case. In essence, Appellant asks for patent protection on the use of encryption on the combination of transaction data and a "unique human identifier" to generate a secure endorsement. At the outset, Examiner notes that Appellant has not disclosed a new method of encrypting data generally. Appellant references known encryption methods in the specification to support the generation of a "unique code" from the data. See, e.g., page 12, lines 2-6; page 17, lines 4-9; page 25, lines 1-6; and page 26, lines 18-25. Therefore, the claim should be seen as directed towards a particular set of data (transaction data and unique human identifier data) which is to be encrypted in a known fashion. The

Appellant has not made of record the particular document referred to at page 12, lines 2-6 and page 25, lines 1-6. At page 17 and 26, the referenced RSA technique is notable. RSA is short for Rivest, Shamir, and Adelman, and this technique is also cited in the Davies reference which was applied against claim 1.

Davies clearly has transaction data, and clearly encrypts this to produce a digital signature. The greatest difficulty with the claim is Appellants' use of the term "unique human identifier". In the Summary of the invention, Appellant has stated that "The unique human identifier 220 may be, for example, a digitized signature, biometric, retinal pattern, or finger print (page 11, lines 23-24)." See Brief, page 3. If the Appellant had actually used this language in the claim, it would have been rejected under 35 USC 112, second paragraph, for indefiniteness. It would be using a broad term and a narrow term in the same claim. Since Appellant used only the broad term in the claim, it cannot be so rejected. This broad/narrow inconsistency has generated much argument during the prosecution. The Examiner still believes the claim should be interpreted broadly, because only the broad language is present in the claim. However, Examiner also decided to take a cautious path in rejecting the claims; Griffith is present because Examiner concluded that reasonable artisans may disagree over whether Davies discloses a "unique human identifier".

Appellant has also stated that "These identifiers are defined as being associated with an individual, are unique to that individual, and are non-transferrable (page 14, lines 18-21)." Examiner found difficulties also with this definition: random numbers or serial numbers

could be associated with an individual, unique to that individual, and non-transferrable by rule or law. A social security number is an example. Moreover, a retinal pattern is, strictly speaking, transferrable by eye transplant, a rare but well known event in the US. Rather than dwell on these difficulties with definitions, Examiner applied Griffith.

In the Appeal Brief, Appellant made a reference and argument to a document not previously cited. Examiner has provided a copy and made this reference ("Tutorial" hereinafter) of record. Tutorial is designed to be a guide for digital signatures published by the American Bar Association, apparently after the filing date of the instant application. Tutorial is an expressly derivative and incomplete teaching of digital signatures. In footnote 18, Tutorial states "For a more thorough examination of properties in a digital signature, see generally Bruce Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C...". This work ("Schneier" hereinafter) is referenced 7 times in Tutorial. For the sake of completeness of the record, Examiner has also included the portion of this classic text on applied cryptography dealing with digital signatures. A copy of Schneier was received by PTO on March 4, 1996, prior to the filing of the instant application. Because Tutorial admits incompleteness, was published after the filing of the instant invention, and derives from Schneier, Examiner believes Schneier to be a more pertinent reference. Finally, Examiner notes that Spies et al. incorporates by reference the first edition of Schneier at column 4, lines 59-65 (the Examiner was unable to locate the first edition, but presumably the basic teachings are closely related).

A. Claims 1-4 and 25-27 have been properly rejected under 35 35 U.S.C. §103(a) as being unpatentable over Davies in view of Griffith

Applicant argues that Davies, which is silent on the use of hash functions, is not applicable to claim 1, which is also silent on hash functions, because Davies implicitly teaches the use of a hash function when generating a digital signature. Examiner respectfully disagrees. For support, Applicant cites Tutorial, a reference which has several shortcomings, as noted above, including the problems of its publication date (Dec. 20, 1996), its derivative nature, and its essential incompleteness. The teaching it is derived from, Schneier, which suffers none of these problems, clearly teaches that a hash value is not inherent in the creation of a digital signature using public key cryptography. Schneier does teach that a hash function may be used if the document is long, because the signature may otherwise be the same length as the document. This is not a problem if the document is short, like the data in a personal check. Schneier also teaches that a hash function is an add-on feature, and wholly unnecessary to the use of public key cryptography in a digital signature. See page 37 of Schneier, "Signing Documents with Public-Key Cryptography". Examiner also finds this line of argument to be at odds with Appellant's own specification. Schneier makes it clear that "hash function" is another word for a "message digest" (page 30). In the text bridging page 11-12 in the specification, Appellant states that "The unique code may be generated by using a checksum algorithm such as CRC or XOR or a message digest from RSA Data Security, Inc., USA (see

BSAFE User's Manual, Version 2.1, p. 42, 1992), or other algorithms with similar characteristics." Thus, even if Davies inherently taught the use of hash functions, which it does not, a fair reading of claim 1 (which is not specific as to the use of hash functions) interpreted in light of the specification (which teaches message digests or hash values in the creation of the unique code) would defeat Appellant's arguments.

Appellant also argues that "it is unreasonable to interpret a customer identity as a unique human identifier". Examiner respectfully disagrees. Applicant makes the argument that since it is possible that some account numbers may not be unique to an individual, that therefore the uniqueness is not applicable. The problem with this line of argument is that any single instance of a customer identity which constituted a unique human identifier would be sufficient as prior art. Thus, even if some account numbers identify a group of people, it is unreasonable to state that all account numbers will identify groups of people. Since some account numbers will be unique to a unique human, and uniquely identify a human, they constitute unique human identifiers. Jack Kent Cooke, owner of the Washington Redskins, was a unique human. Any number identifying him and no other human is a unique human identifier. Were he a customer writing a check according to Davies, the "customer identity" would be a unique human identifier.

That said, Appellant is missing a larger point. In the check of Davies is a Customer Identity 5, a Customer public key 6, which is also unique to the customer, and therefore also a unique human identifier, and a customer private key, which is used to create 16. The

customer private key is unique to a particular customer. If it was not, then another customer could forge documents. Uniqueness of the private (secret) key is axiomatic in public key cryptography. It reliably ties the creation of a digital signature with a unique customer, which as explained above, is in many (most) instances a unique human.

Appellant argues that the unique identifier is defined as "associated with an individual, is unique to the individual, and non-transferrable." Appellant further argues that a customer identity is transferrable. Examiner respectfully disagrees. In the normal sense of the word, a customer identity is not normally transferrable. For example, I cannot transfer my identity to someone else for the purpose of writing checks. This would typically be viewed as fraud. The laws of the US do not allow the transfer of identity for the purposes of making transactions such as writing checks. I, Mark Tremblay, a unique human, could not transfer my identity to my supervisor, Donald Hajec, so that he could write checks using my identity. I could also not transfer my identity to any one of the other Mark Tremblays living in the United States for this purpose. When I write a check using my personal bank account, a unique human (me) is identified by the check to be the payor. Anything else would turn the concept of identity on its head.

Appellant also argues that even if customer identity 5 of Davies could reasonably be interpreted as a unique human identifier as claimed, that element of Fig. 1 is not used in creating signature 16. Examiner respectfully disagrees. As noted above, a secret key is used to create the digital signature 16. The secret key is essentially a unique human identifier,

because it is directly related to a customer identity by the public key. Anyone knowing a public key will have a unique human identifier because the public key must be related to the customer identity on a 1:1 basis. The receiver of the public key would merely look up the key in a registry, and find the customer identity. Likewise, a private or secret key is also associated with a customer identity on a 1:1 basis, and is as non-transferrable as a customer identity. The secret key being unique to a particular individual is the very basis of the digital signature process. If it did not uniquely identify an individual, the digital signature would be meaningless!

Even so, the question here is not anticipation. It is obviousness over the teachings of Davies and Griffith. The Examiner admitted that Davies does not teach a "unique human identifier" in the narrow sense: e.g. a biometric. Griffith provides a clear teaching of the use of a biometric encrypted using public key cryptography to thwart fraud. Thus, in the same way that a unique human adds a biometric (a handwritten signature) to a paper check to thwart fraud, a biometric such as a digitized signature, fingerprint, or retinal scan may be added to a digital check to thwart fraud. In the same way that a person stealing a checkbook from another could fraudulently write checks using the checkbook, but would have difficulty forging the biometric, so could a person stealing a smart card use the smart card to fraudulently create checks. Even if a PIN number were used to protect the card, that too can be stolen, as is well known in the art. If a PIN were perfect or wholly sufficient protection, why would Griffith, which discusses the use of a PIN in the background of the invention, also teach the use of a

biometric? The addition of the biometric taught by Griffith to the check of Davies is as natural a combination as a handwritten signature to a paper check. It would have been obvious at the time the invention was made to make this combination.

Appellant argues that Griffith teaches a "private key cryptosystem", and is not applicable to the claims at issue or combinable with Davies. Examiner respectfully disagrees. First, the claims at issue do not specify any particular cryptosystem, public or symmetric. Second, even a reading of the abstract of Griffith reveals that both symmetric and public key crpytography are used in Griffith. Finally, these arguments are immaterial to the basic combination at issue. Griffith provides the teaching that a biometric may be used to enhance the creation of an unforgeable digital signature. Virtually anything can be forged; it is only a question of how difficult it is to do so. Griffith adds a biometric to a digital signature, raising the level of difficulty from extremely difficult to even more extremely difficult.

Appellant also argues that "...Davies expressly chose to create a system using a public key encryption and not secret key encryption." (page 17). Examiner respectfully disagrees with this statement. Davies, and any public key system, involves the use of secret or private keys as well as public keys. Much of Appellant's argument in the full paragraph on page 17 does not make sense. Appellant appears to draw a stark contrast between the system used by Davies and the system used by Davies.

The Appellant's conclusion that there is no motivation to combine the biometric security of Griffith with the system taught by Davies simply is not supported by Appellant's arguments.

2. Group II: Claims 2 and 4

Appellant argues that Davies does not meet the limitations of claim 2, because in doing so, Davies would make public the private key. Examiner respectfully disagrees. While Applicant is correct in asserting that if Davies provided the private key as part of a whole representation of the secure endorsed transaction, this does not comport with the combination made by Examiner. Clearly, in the instance of claim 2, Examiner cannot look to the private key as a unique human identifier. Examiner relied upon Griffith for a teaching of a biometric which would constitute a unique human identifier. Davies clearly teaches a whole representation of the secure endorsed transaction in figure 1. The addition of the biometric taught by Griffith adds to the security of this endorsement, and this would be provided as an input in the transaction data in the way it is taught as an input to the transaction data in Griffith. There is no necessity to keep the biometric secret, any more than there is a reason to keep the signature on a check secret. It may be used as an augmentation to securely identify the presence of an individual at the time the document was created. Essentially, Appellant's arguments focus on the novel details of Griffith, and not the obvious features relied upon by the Examiner. Thus, Appellants focus on the details has missed the bigger picture.

B. Claims 5-23 and 29-31 have been properly rejected under 35 U.S.C. §103(a) as being unpatentable over Davies in view of Griffith and further in view of Spies.

3. Group III: Claims 5-7 and 20-23

Appellant repeats some of the arguments addressed above. Appellant also argues that Spies suffers from the same defects as Davies and Griffith, in that Spies uses both symmetric and public key cryptography, and therefore has the same problem when sending a message including a code and the data used to generate that code. Examiner respectfully disagrees. Davies clearly has no problem sending a digital signature together with the information used to create that code, with the exception of the secret key. Appellant also teaches the use of RSA public key cryptography in the specification, and likewise could not send the secret key. That is not really the issue. The issue is whether the biometric would be sent along with the other transaction data. Examiner has addressed this above. The biometric, like the customer identity of Davies and like the signature on a paper check, may be sent along with the other data used to create the secure authorized transaction, without the need to send the private key. Appellant's own system would not send the private key.

In addition, even if there was a requirement that the secret key be sent with the transaction data, Spies provides a way to do this, by encrypting the keys using another public/private or symmetric algorithm. See e.g. the abstract ("...the CSP prevents exposure of the encryption keys in a non-encrypted form...") and figure 4. Applicants arguments therefore do not show that the combination fails to meet the claim limitations.

4. Group IV: Claims 8-10

The Appellant has essentially repeated arguments made previously. The encrypted code, i.e. the digital signature, would be included together with the data used to generate the code. This is precisely what Davies teaches, and these teachings are augmented in an obvious manner by Griffith and Spies. Spies further teaches a method for preventing the exposure of the encryption keys in a non-encrypted form.

5. Group V: Claims 11-14 and 29-31

Appellant argues that the combined teachings do not deal with the signature of the data by a first party using a received public key, and by a second party using a private key corresponding to the received public key. Examiner respectfully disagrees. The Appellant has argued against Davies by itself, and then asserted without substantial analysis that Griffith and Spies fail to cure the supposed deficiencies of Davies. Clearly, if Examiner thought Davies was sufficient, Griffith and Spies would have been left out of the rejection.

The use of a public key to send data to one individual only is one of the basic teachings of public key cryptography. To make an analogy, if Appellant had applied for a method of providing a stock price over a telephone, and claimed in addition that the phone number of the providing party was first dialed by the quote receiving party, the additional dialing limitation would not make the claim more patentable, because it is a part and parcel of using a telephone, so basic that any skilled artisan would understand. So if the first party wants to send the transaction data and a unique human identifier to a second party, such that

only the second party may read the transaction data and the unique human identifier, the first party may use the public key of the second party. See either Tutorial or Schneier. Note that Spies has incorporated by reference the first edition of Schneier. Note also that the discussions found in Spies, and arguably in Davies and Griffith, assumes that the reader is familiar with cryptography (see Spies, column 4, lines 59-60). Because Spies deals with a network, the teachings regarding the use of the public key are provided more explicitly. See e.g. column 8, lines 49-60, and elsewhere in Spies.

6. Group VI: Claims 15-18

Appellant argues that Davies does not teach the verification process claimed, for reasons argued previously, and that Griffith and Spies fail to cure the asserted deficiencies of Davies. The Examiner respectfully disagrees. Again, Davies is not limited to hash values, as described above. Davies provides both the transaction data and the signed transaction data in the endorsed document. The combination with Griffith adds a biometric for enhanced fraud resistance. The verification process is basic to public key cryptography, and is taught in both Spies and Schneier, incorporated by reference. In fact, claim 15 is a semantic variation on claim 11. If the transaction data is endorsed using a public key as recited in claim 11, then the decryption process using the secret key corresponding to the public key is the verification process that will allow verification and comparison. This directly follows from the limitations of claim 11, and it is taught by Spies, as described above. Schneier makes plain that the

decryption process verifies digital signature when the decryption restores the original message, and does not necessarily involve the use of a hash function.

7. Group VII: Claim 19

Appellant argues that the rejection of claim 16 is invalid under this heading. The reasons for doing so are not understood, since claim 16 was argued previously as standing or falling with claim 15, and claim 15 has been addressed. Nevertheless, claim 16 was meaningfully treated in the combined teachings of the rejection. The combined teachings are part of the rejection without the Examiner copying or restating everything in the teachings in the statement of the rejection; in other words, the combined teachings are incorporated by reference into the rejection.

The arguments for this group do not appear to be distinct from the previously made arguments. Again, this claim is a variation on the type of transaction claimed in claim 11. The verification procedure is basic to public key cryptography. If a transaction is endorsed by a public key, then the verification procedure is substantially the one claimed in claim 19.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Mark Tremblay
July 17, 2000

Conferees:

Karl Frech

Donald Hajec

FORM PTO-892	U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE	SERIAL NO.	GROUP ART UNIT	ATTACHMENT TO PAPER NO.	
NOTICE OF REFERENCES CITED		08697421	2876		24
		APPLICANT(S) Movalli et al.			

U.S. PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	NAME	CLASS	SUB-CLASS	FILING DATE
	A						
	B						
	C						
	D						
	E						
	F						
	G						
	H						
	I						
	J						
	K						

FOREIGN PATENT DOCUMENTS

*		DOCUMENT NO.	DATE	COUNTRY	NAME	CLASS	SUB-CLASS
	L						
	M						
	N						
	O						
	P						
	Q						

OTHER REFERENCES (Including Author, Title, Date, Pertinent Pages, Etc.)

	R	Schneier, Bruce Applied Cryptography, Second Edition, John Wiley and Sons (1996) pages 30-44
	S	Digital Singature Guidelines Tutorial, published by the ABA, 1996, pages 1-8.
	T	
	U	

EXAMINER	DATE	
Mark Tremblay	July 17, 2000	Form892ccs2106b

* A copy of this reference is not being furnished with this office action.
(See Manual of Patent Examining Procedure, section 707.05(a).)

APPLIED CRYPTOGRAPHY, SECOND EDITION

PROTOCOLS, ALGORITHMS, AND SOURCE CODE IN C

BRUCE SCHNEIER

John Wiley & Sons, Inc.
New York ◦ Chichester ◦ Brisbane ◦ Toronto ◦ Singapore

Stc

Publisher: Katherine Schowalter
Editor: Phil Sutherland
Assistant Editor: Allison Roarty
Managing Editor: Robert Aronds
Text Design & Composition: North Market Street Graphics

This text is printed on acid-free paper.

Published by John Wiley & Sons, Inc.

Library of Congress Cataloging-in-Publication Data:

Schneier, Bruce
Applied Cryptography Second Edition : protocols, algorithms, and source code in C / Bruce Schneier.
p. cm.
Includes bibliographical references (p. 675).
ISBN 0-471-12845-7 (cloth : acid-free paper). — ISBN 0-471-11709-9 (paper : acid-free paper)
1. Computer security. 2. Telecommunication—Security measures. 3. Cryptography. I. Title.
QA76.9.A25S35 1996
005.8'2—dc20 95-12398
CIP

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

he is with the one-way function.) For public-key cryptography, we need something else (although there are cryptographic applications for one-way functions—see Section 3.2).

A **trapdoor one-way function** is a special type of one-way function, one with a secret trapdoor. It is easy to compute in one direction and hard to compute in the other direction. But, if you know the secret, you can easily compute the function in the other direction. That is, it is easy to compute $f(x)$ given x, and hard to compute x given $f(x)$. However, there is some secret information, y, such that given $f(x)$ and y it is easy to compute x.

Taking a watch apart is a good example of a trap-door one-way function. It is easy to disassemble a watch into hundreds of minuscule pieces. It is very difficult to put those tiny pieces back together into a working watch. However, with the secret information—the assembly instructions of the watch—it is much easier to put the watch back together.

2.4 ONE-WAY HASH FUNCTIONS

A **one-way hash function** has many names: compression function, contraction function, message digest, fingerprint, cryptographic checksum, message integrity check (MIC), and manipulation detection code (MDC). Whatever you call it, it is central to modern cryptography. One-way hash functions are another building block for many protocols.

Hash functions have been used in computer science for a long time. A hash function is a function, mathematical or otherwise, that takes a variable-length input string (called a **pre-image**) and converts it to a fixed-length (generally smaller) output string (called a **hash value**). A simple hash function would be a function that takes pre-image and returns a byte consisting of the XOR of all the input bytes.

The point here is to fingerprint the pre-image: to produce a value that indicates whether a candidate pre-image is likely to be the same as the real pre-image. Because hash functions are typically many-to-one, we cannot use them to determine with certainty that the two strings are equal, but we can use them to get a reasonable assurance of accuracy.

A one-way hash function is a hash function that works in one direction: It is easy to compute a hash value from pre-image, but it is hard to generate a pre-image that hashes to a particular value. The hash function previously mentioned is not one-way: Given a particular byte value, it is trivial to generate a string of bytes whose XOR is that value. You can't do that with a one-way hash function. A good one-way hash function is also **collision-free**: It is hard to generate two pre-images with the same hash value.

The hash function is public; there's no secrecy to the process. The security of a one-way hash function is its one-wayness. The output is not dependent on the input in any discernible way. A single bit change in the pre-image changes, on the average, half of the bits in the hash value. Given a hash value, it is computationally unfeasible to find a pre-image that hashes to that value.

Think of it as a way of fingerprinting files. If you want to verify that someone has a particular file (that you also have), but you don't want him to send it to you, then ask him for the hash value. If he sends you the correct hash value, then it is almost certain that he has that file. This is particularly useful in financial transactions, where you don't want a withdrawal of $100 to turn into a withdrawal of $1000 somewhere in the network. Normally, you would use a one-way hash function without a key, so that anyone can verify the hash. If you want only the recipient to be able to verify the hash, then read the next section.

Message Authentication Codes

A **message authentication code** (MAC), also known as a data authentication code (DAC), is a one-way hash function with the addition of a secret key (see Section 18.14). The hash value is a function of both the pre-image and the key. The theory is exactly the same as hash functions, except only someone with the key can verify the hash value. You can create a MAC out of a hash function or a block encryption algorithm; there are also dedicated MACs.

2.5 Communications Using Public-Key Cryptography

Think of a symmetric algorithm as a safe. The key is the combination. Someone with the combination can open the safe, put a document inside, and close it again. Someone else with the combination can open the safe and take the document out. Anyone without the combination is forced to learn safecracking.

In 1976, Whitfield Diffie and Martin Hellman changed that paradigm of cryptography forever [496]. (The NSA has claimed knowledge of the concept as early as 1966, but has offered no proof.) They described **public-key cryptography**. They used two different keys—one public and the other private. It is computationally hard to deduce the private key from the public key. Anyone with the public key can encrypt a message but not decrypt it. Only the person with the private key can decrypt the message. It is as if someone turned the cryptographic safe into a mailbox. Putting mail in the mailbox is analogous to encrypting with the public key; anyone can do it. Just open the slot and drop it in. Getting mail out of a mailbox is analogous to decrypting with the private key. Generally it's hard; you need welding torches. However, if you have the secret (the physical key to the mailbox), it's easy to get mail out of a mailbox.

Mathematically, the process is based on the trap-door one-way functions previously discussed. Encryption is the easy direction. Instructions for encryption are the public key; anyone can encrypt a message. Decryption is the hard direction. It's made hard enough that people with Cray computers and thousands (even millions) of years couldn't decrypt the message without the secret. The secret, or trapdoor, is the private key. With that secret, decryption is as easy as encryption.

This is how Alice can send a message to Bob using public-key cryptography:

(1) Alice and Bob agree on a public-key cryptosystem.

(2) Bob sends Alice his public key.
(3) Alice encrypts her message using Bob's public key and sends it to Bob.
(4) Bob decrypts Alice's message using his private key.

Notice how public-key cryptography solves the key-management problem with symmetric cryptosystems. Before, Alice and Bob had to agree on a key in secret. Alice could choose one at random, but she still had to get it to Bob. She could hand it to him sometime beforehand, but that requires foresight. She could send it to him by secure courier, but that takes time. Public-key cryptography makes it easy. With no prior arrangements, Alice can send a secure message to Bob. Eve, listening in on the entire exchange, has Bob's public key and a message encrypted in that key, but cannot recover either Bob's private key or the message.

More commonly, a network of users agrees on a public-key cryptosystem. Every user has his or her own public key and private key, and the public keys are all published in a database somewhere. Now the protocol is even easier:

(1) Alice gets Bob's public key from the database.
(2) Alice encrypts her message using Bob's public key and sends it to Bob.
(3) Bob then decrypts Alice's message using his private key.

In the first protocol, Bob had to send Alice his public key before she could send him a message. The second protocol is more like traditional mail. Bob is not involved in the protocol until he wants to read his message.

Hybrid Cryptosystems

The first public-key algorithms became public at the same time that DES was being discussed as a proposed standard. This resulted in some partisan politics in the cryptographic community. As Diffie described it [494]:

> The excitement public key cryptosystems provoked in the popular and scientific press was not matched by corresponding acceptance in the cryptographic establishment, however. In the same year that public key cryptography was discovered, the National Security Agency (NSA), proposed a conventional cryptographic system, designed by International Business Machines (IBM), as a federal *Data Encryption Standard* (DES). Marty Hellman and I criticized the proposal on the ground that its key was too small, but manufacturers were gearing up to support the proposed standard and our criticism was seen by many as an attempt to disrupt the standards-making process to the advantage of our own work. Public key cryptography in its turn was attacked, in sales literature [1125] and technical papers [849,1159] alike, more as though it were a competing product than a recent research discovery. This, however, did not deter the NSA from claiming its share of the credit. Its director, in the words of the *Encyclopedia Britannica* [1461], pointed out that "two-key cryptography had been discovered at the agency a decade earlier," although no evidence for this claim was ever offered publicly.

In the real world, public-key algorithms are not a substitute for symmetric algorithms. They are not used to encrypt messages; they are used to encrypt keys. There are two reasons for this:

1. Public-key algorithms are slow. Symmetric algorithms are generally at least 1000 times faster than public-key algorithms. Yes, computers are getting faster and faster, and in 15 years computers will be able to do public-key cryptography at speeds comparable to symmetric cryptography today. But bandwidth requirements are also increasing, and there will always be the need to encrypt data faster than public-key cryptography can manage.
2. Public-key cryptosystems are vulnerable to chosen-plaintext attacks. If $C = E(P)$, when P is one plaintext out of a set of n possible plaintexts, then a cryptanalyst only has to encrypt all n possible plaintexts and compare the results with C (remember, the encryption key is public). He won't be able to recover the decryption key this way, but he will be able to determine P.

A chosen-plaintext attack can be particularly effective if there are relatively few possible encrypted messages. For example, if P were a dollar amount less than $1,000,000, this attack would work; the cryptanalyst tries all million possible dollar amounts. (Probabilistic encryption solves the problem; see Section 23.15.) Even if P is not as well-defined, this attack can be very effective. Simply knowing that a ciphertext does not correspond to a particular plaintext can be useful information. Symmetric cryptosystems are not vulnerable to this attack because a cryptanalyst cannot perform trial encryptions with an unknown key.

In most practical implementations public-key cryptography is used to secure and distribute **session keys**; those session keys are used with symmetric algorithms to secure message traffic [879]. This is sometimes called a **hybrid cryptosystem**.

(1) Bob sends Alice his public key.

(2) Alice generates a random session key, K, encrypts it using Bob's public key, and sends it to Bob.

$$E_B(K)$$

(3) Bob decrypts Alice's message using his private key to recover the session key.

$$D_B(E_B(K)) = K$$

(4) Both of them encrypt their communications using the same session key.

Using public-key cryptography for key distribution solves a very important key-management problem. With symmetric cryptography, the data encryption key sits around until it is used. If Eve ever gets her hands on it, she can decrypt messages encrypted with it. With the previous protocol, the session key is created when it is needed to encrypt communications and destroyed when it is no longer needed. This drastically reduces the risk of compromising the session key. Of course, the private

key is vulnerable to compromise, but it is at less risk because it is only used once per communication to encrypt a session key. This is further discussed in Section 3.1.

Merkle's Puzzles

Ralph Merkle invented the first construction of public-key cryptography. In 1974 he registered for a course in computer security at the University of California, Berkeley, taught by Lance Hoffman. His term paper topic, submitted early in the term, addressed the problem of "Secure Communication over Insecure Channels" [1064]. Hoffman could not understand Merkle's proposal and eventually Merkle dropped the course. He continued to work on the problem, despite continuing failure to make his results understood.

Merkle's technique was based on "puzzles" that were easier to solve for the sender and receiver than for an eavesdropper. Here's how Alice sends an encrypted message to Bob without first having to exchange a key with him.

(1) Bob generates 2^{20}, or about a million, messages of the form: "This is puzzle number x. This is the secret key number y," where x is a random number and y is a random secret key. Both x and y are different for each message. Using a symmetric algorithm, he encrypts each message with a different 20-bit key and sends them all to Alice.

(2) Alice chooses one message at random and performs a brute-force attack to recover the plaintext. This is a large, but not impossible, amount of work.

(3) Alice encrypts her secret message with the key she recovered and some symmetric algorithm, and sends it to Bob along with x.

(4) Bob knows which secret key y he encrypts in message x, so he can decrypt the message.

Eve can break this system, but she has to do far more work than either Alice or Bob. To recover the message in step (3), she has to perform a brute-force attack against each of Bob's 2^{20} messages in step (1); this attack has a complexity of 2^{40}. The x values won't help Eve either; they were assigned randomly in step (1). In general, Eve has to expend approximately the square of the effort that Alice expends.

This n to n^2 advantage is small by cryptographic standards, but in some circumstances it may be enough. If Alice and Bob can try ten thousand keys per second, it will take them a minute each to perform their steps and another minute to communicate the puzzles from Bob to Alice on a 1.544 MB link. If Eve had comparable computing facilities, it would take her about a year to break the system. Other algorithms are even harder to break.

2.6 DIGITAL SIGNATURES

Handwritten signatures have long been used as proof of authorship of, or at least agreement with, the contents of a document. What is it about a signature that is so compelling [1392]?

1. The signature is authentic. The signature convinces the document's recipient that the signer deliberately signed the document.
2. The signature is unforgeable. The signature is proof that the signer, and no one else, deliberately signed the document.
3. The signature is not reusable. The signature is part of the document; an unscrupulous person cannot move the signature to a different document.
4. The signed document is unalterable. After the document is signed, it cannot be altered.
5. The signature cannot be repudiated. The signature and the document are physical things. The signer cannot later claim that he or she didn't sign it.

In reality, none of these statements about signatures is completely true. Signatures can be forged, signatures can be lifted from one piece of paper and moved to another, and documents can be altered after signing. However, we are willing to live with these problems because of the difficulty in cheating and the risk of detection.

We would like to do this sort of thing on computers, but there are problems. First, computer files are trivial to copy. Even if a person's signature were difficult to forge (a graphical image of a written signature, for example), it would be easy to cut and paste a valid signature from one document to another document. The mere presence of such a signature means nothing. Second, computer files are easy to modify after they are signed, without leaving any evidence of modification.

Signing Documents with Symmetric Cryptosystems and an Arbitrator

Alice wants to sign a digital message and send it to Bob. With the help of Trent and a symmetric cryptosystem, she can.

Trent is a powerful, trusted arbitrator. He can communicate with both Alice and Bob (and everyone else who may want to sign a digital document). He shares a secret key, K_A, with Alice, and a different secret key, K_B, with Bob. These keys have been established long before the protocol begins and can be reused multiple times for multiple signings.

(1) Alice encrypts her message to Bob with K_A and sends it to Trent.
(2) Trent decrypts the message with K_A.
(3) Trent takes the decrypted message and a statement that he has received this message from Alice, and encrypts the whole bundle with K_B.
(4) Trent sends the encrypted bundle to Bob.
(5) Bob decrypts the bundle with K_B. He can now read both the message and Trent's certification that Alice sent it.

How does Trent know that the message is from Alice and not from some imposter? He infers it from the message's encryption. Since only he and Alice share their secret key, only Alice could encrypt a message using it.

Is this as good as a paper signature? Let's look at the characteristics we want:

1. This signature is authentic. Trent is a trusted arbitrator and Trent knows that the message came from Alice. Trent's certification serves as proof to Bob.
2. This signature is unforgeable. Only Alice (and Trent, but everyone trusts him) knows K_A, so only Alice could have sent Trent a message encrypted with K_A. If someone tried to impersonate Alice, Trent would have immediately realized this in step (2) and would not certify its authenticity.
3. This signature is not reusable. If Bob tried to take Trent's certification and attach it to another message, Alice would cry foul. An arbitrator (it could be Trent or it could be a completely different arbitrator with access to the same information) would ask Bob to produce both the message and Alice's encrypted message. The arbitrator would then encrypt the message with K_A and see that it did not match the encrypted message that Bob gave him. Bob, of course, could not produce an encrypted message that matches because he does not know K_A.
4. The signed document is unalterable. Were Bob to try to alter the document after receipt, Trent could prove foul play in exactly the same manner just described.
5. The signature cannot be repudiated. Even if Alice later claims that she never sent the message, Trent's certification says otherwise. Remember, Trent is trusted by everyone; what he says is true.

If Bob wants to show Carol a document signed by Alice, he can't reveal his secret key to her. He has to go through Trent again:

(1) Bob takes the message and Trent's statement that the message came from Alice, encrypts them with K_B, and sends them back to Trent.

(2) Trent decrypts the bundle with K_B.

(3) Trent checks his database and confirms that the original message came from Alice.

(4) Trent re-encrypts the bundle with the secret key he shares with Carol, K_C, and sends it to Carol.

(5) Carol decrypts the bundle with K_C. She can now read both the message and Trent's certification that Alice sent it.

These protocols work, but they're time-consuming for Trent. He must spend his days decrypting and encrypting messages, acting as the intermediary between every pair of people who want to send signed documents to one another. He must keep a database of messages (although this can be avoided by sending the recipient a copy of the sender's encrypted message). He is a bottleneck in any communications system, even if he's a mindless software program.

Harder still is creating and maintaining someone like Trent, someone that everyone on the network trusts. Trent has to be infallible; if he makes even one mistake in a million signatures, no one is going to trust him. Trent has to be completely secure. If his database of secret keys ever got out or if someone managed to modify his programming, everyone's signatures would be completely useless. False documents purported to be signed years ago could appear. Chaos would result. Governments would collapse. Anarchy would reign. This might work in theory, but it doesn't work very well in practice.

Digital Signature Trees

Ralph Merkle proposed a digital signature scheme based on secret-key cryptography, producing an infinite number of one-time signatures using a tree structure [1067,1068]. The basic idea of this scheme is to place the root of the tree in some public file, thereby authenticating it. The root signs one message and authenticates its sub-nodes in the tree. Each of these nodes signs one message and authenticates its sub-nodes, and so on.

Signing Documents with Public-Key Cryptography

There are public-key algorithms that can be used for digital signatures. In some algorithms—RSA is an example (see Section 19.3)—either the public key or the private key can be used for encryption. Encrypt a document using your private key, and you have a secure digital signature. In other cases—DSA is an example (see Section 20.1)—there is a separate algorithm for digital signatures that cannot be used for encryption. This idea was first invented by Diffie and Hellman [496] and further expanded and elaborated on in other texts [1282,1328,1024,1283,426]. See [1099] for a good survey of the field.

The basic protocol is simple:

(1) Alice encrypts the document with her private key, thereby signing the document.
(2) Alice sends the signed document to Bob.
(3) Bob decrypts the document with Alice's public key, thereby verifying the signature.

This protocol is far better than the previous one. Trent is not needed to either sign or verify signatures. (He is needed to certify that Alice's public key is indeed her public key.) The parties do not even need Trent to resolve disputes: If Bob cannot perform step (3), then he knows the signature is not valid.

This protocol also satisfies the characteristics we're looking for:

1. The signature is authentic; when Bob verifies the message with Alice's public key, he knows that she signed it.
2. The signature is unforgeable; only Alice knows her private key.
3. The signature is not reusable; the signature is a function of the document and cannot be transferred to any other document.

4. The signed document is unalterable; if there is any alteration to the document, the signature can no longer be verified with Alice's public key.
5. The signature cannot be repudiated. Bob doesn't need Alice's help to verify her signature.

Signing Documents and Timestamps

Actually, Bob can cheat Alice in certain circumstances. He can reuse the document and signature together. This is no problem if Alice signed a contract (what's another copy of the same contract, more or less?), but it can be very exciting if Alice signed a digital check.

Let's say Alice sends Bob a signed digital check for $100. Bob takes the check to the bank, which verifies the signature and moves the money from one account to the other. Bob, who is an unscrupulous character, saves a copy of the digital check. The following week, he again takes it to the bank (or maybe to a different bank). The bank verifies the signature and moves the money from one account to the other. If Alice never balances her checkbook, Bob can keep this up for years.

Consequently, digital signatures often include timestamps. The date and time of the signature are attached to the message and signed along with the rest of the message. The bank stores this timestamp in a database. Now, when Bob tries to cash Alice's check a second time, the bank checks the timestamp against its database. Since the bank already cashed a check from Alice with the same timestamp, the bank calls the police. Bob then spends 15 years in Leavenworth prison reading up on cryptographic protocols.

Signing Documents with Public-Key Cryptography and One-Way Hash Functions

In practical implementations, public-key algorithms are often too inefficient to sign long documents. To save time, digital signature protocols are often implemented with one-way hash functions [432,433]. Instead of signing a document, Alice signs the hash of the document. In this protocol, both the one-way hash function and the digital signature algorithm are agreed upon beforehand.

(1) Alice produces a one-way hash of a document.
(2) Alice encrypts the hash with her private key, thereby signing the document.
(3) Alice sends the document and the signed hash to Bob.
(4) Bob produces a one-way hash of the document that Alice sent. He then, using the digital signature algorithm, decrypts the signed hash with Alice's public key. If the signed hash matches the hash he generated, the signature is valid.

Speed increases drastically and, since the chances of two different documents having the same 160-bit hash are only one in 2^{160}, anyone can safely equate a signature of the hash with a signature of the document. If a non-one-way hash function were

used, it would be an easy matter to create multiple documents that hashed to the same value, so that anyone signing a particular document would be duped into signing a multitude of documents.

This protocol has other benefits. First, the signature can be kept separate from the document. Second, the recipient's storage requirements for the document and signature are much smaller. An archival system can use this type of protocol to verify the existence of documents without storing their contents. The central database could just store the hashes of files. It doesn't have to see the files at all; users submit their hashes to the database, and the database timestamps the submissions and stores them. If there is any disagreement in the future about who created a document and when, the database could resolve it by finding the hash in its files. This system has vast implications concerning privacy: Alice could copyright a document but still keep the document secret. Only if she wished to prove her copyright would she have to make the document public. (See Section 4.1).

Algorithms and Terminology

There are many digital signature algorithms. All of them are public-key algorithms with secret information to sign documents and public information to verify signatures. Sometimes the signing process is called **encrypting with a private key** and the verification process is called **decrypting with a public key**. This is misleading and is only true for one algorithm, RSA. And different algorithms have different implementations. For example, one-way hash functions and timestamps sometimes add extra steps to the process of signing and verifying. Many algorithms can be used for digital signatures, but not for encryption.

In general, I will refer to the signing and verifying processes without any details of the algorithms involved. Signing a message with private key K is:

$$S_K(M)$$

and verifying a signature with the corresponding public key is:

$$V_K(M)$$

The bit string attached to the document when signed (in the previous example, the one-way hash of the document encrypted with the private key) will be called the **digital signature**, or just the **signature**. The entire protocol, by which the receiver of a message is convinced of the identity of the sender and the integrity of the message, is called authentication. Further details on these protocols are in Section 3.2.

Multiple Signatures

How could Alice and Bob sign the same digital document? Without one-way hash functions, there are two options. One is that Alice and Bob sign separate copies of the document itself. The resultant message would be over twice the size of the original document. The second is that Alice signs the document first and then Bob signs Alice's signature. This works, but it is impossible to verify Alice's signature without also verifying Bob's.

With one-way hash functions, multiple signatures are easy:

(1) Alice signs the hash of the document.
(2) Bob signs the hash of the document.
(3) Bob sends his signature to Alice.
(4) Alice sends the document, her signature, and Bob's signature to Carol.
(5) Carol verifies both Alice's signature and Bob's signature.

Alice and Bob can do steps (1) and (2) either in parallel or in series. In step (5), Carol can verify one signature without having to verify the other.

Nonrepudiation and Digital Signatures

Alice can cheat with digital signatures and there's nothing that can be done about it. She can sign a document and then later claim that she did not. First, she signs the document normally. Then, she anonymously publishes her private key, conveniently loses it in a public place, or just pretends to do either one. Alice then claims that her signature has been compromised and that others are using it, pretending to be her. She disavows signing the document and any others that she signed using that private key. This is called repudiation.

Timestamps can limit the effects of this kind of cheating, but Alice can always claim that her key was compromised earlier. If Alice times things well, she can sign a document and then successfully claim that she didn't. This is why there is so much talk about private keys buried in tamper-resistant modules—so that Alice can't get at hers and abuse it.

Although nothing can be done about this possible abuse, one can take steps to guarantee that old signatures are not invalidated by actions taken in disputing new ones. (For example, Alice could "lose" her key to keep from paying Bob for the junk car he sold her yesterday and, in the process, invalidate her bank account.) The solution is for the receiver of a signed document to have it timestamped [453].

The general protocol is given in [28]:

(1) Alice signs a message.
(2) Alice generates a header containing some identifying information. She concatenates the header with the signed message, signs that, and sends it to Trent.
(3) Trent verifies the outside signature and confirms the identifying information. He adds a timestamp to Alice's signed message and the identifying information. Then he signs it all and sends it to both Alice and Bob.
(4) Bob verifies Trent's signature, the identifying information, and Alice's signature.
(5) Alice verifies the message Trent sent to Bob. If she did not originate the message, she speaks up quickly.

Another scheme uses Trent after the fact [209]. After receiving a signed message, Bob can send a copy to Trent for verification. Trent can attest to the validity of Alice's signature.

Applications of Digital Signatures

One of the earliest proposed applications of digital signatures was to facilitate the verification of nuclear test ban treaties [1454,1467]. The United States and the Soviet Union (anyone remember the Soviet Union?) permitted each other to put seismometers on the other's soil to monitor nuclear tests. The problem was that each country needed to assure itself that the host nation was not tampering with the data from the monitoring nation's seismometers. Simultaneously, the host nation needed to assure itself that the monitor was sending only the specific information needed for monitoring.

Conventional authentication techniques can solve the first problem, but only digital signatures can solve both problems. The host nation can read, but not alter, data from the seismometer, and the monitoring nation knows that the data has not been tampered with.

2.7 Digital Signatures with Encryption

By combining digital signatures with public-key cryptography, we develop a protocol that combines the security of encryption with the authenticity of digital signatures. Think of a letter from your mother: The signature provides proof of authorship and the envelope provides privacy.

(1) Alice signs the message with her private key.

$$S_A(M)$$

(2) Alice encrypts the signed message with Bob's public key and sends it to Bob.

$$E_B(S_A(M))$$

(3) Bob decrypts the message with his private key.

$$D_B(E_B(S_A(M))) = S_A(M)$$

(4) Bob verifies with Alice's public key and recovers the message.

$$V_A(S_A(M)) = M$$

Signing before encrypting seems natural. When Alice writes a letter, she signs it and then puts it in an envelope. If she put the letter in the envelope unsigned and then signed the envelope, then Bob might worry if the letter hadn't been covertly replaced. If Bob showed to Carol Alice's letter and envelope, Carol might accuse Bob of lying about which letter arrived in which envelope.

In electronic correspondence as well, signing before encrypting is a prudent practice [48]. Not only is it more secure—an adversary can't remove a signature from an encrypted message and add his own—but there are legal considerations: If the text

to be signed is not visible to the signer when he affixes his signature, then the signature may have little legal force [1312]. And there are some cryptanalytic attacks against this technique with RSA signatures (see Section 19.3).

There's no reason Alice has to use the same public-key/private-key key pair for encrypting and signing. She can have two key pairs: one for encryption and the other for signatures. Separation has its advantages: she can surrender her encryption key to the police without compromising her signature, one key can be escrowed (see Section 4.13) without affecting the other, and the keys can have different sizes and can expire at different times.

Of course, timestamps should be used with this protocol to prevent reuse of messages. Timestamps can also protect against other potential pitfalls, such as the one described below.

Resending the Message as a Receipt

Consider an implementation of this protocol, with the additional feature of confirmation messages. Whenever Bob receives a message, he returns it as a confirmation of receipt.

(1) Alice signs a message with her private key, encrypts it with Bob's public key, and sends it to Bob.

$$E_B[S_A[M]]$$

(2) Bob decrypts the message with his private key and verifies the signature with Alice's public key, thereby verifying that Alice signed the message and recovering the message.

$$V_A[D_B[E_B[S_A[M]]]] = M$$

(3) Bob signs the message with his private key, encrypts it with Alice's public key, and sends it back to Alice.

$$E_A[S_B[M]]$$

(4) Alice decrypts the message with her private key and verifies the signature with Bob's public key. If the resultant message is the same one she sent to Bob, she knows that Bob received the message accurately.

If the same algorithm is used for both encryption and digital-signature verification there is a possible attack [506]. In these cases, the digital signature operation is the inverse of the encryption operation: $V_X = E_X$ and $S_X = D_X$.

Assume that Mallory is a legitimate system user with his own public and private key. Now, let's watch as he reads Bob's mail. First, he records Alice's message to Bob in step (1). Then, at some later time, he sends that message to Bob, claiming that it came from him (Mallory). Bob thinks that it is a legitimate message from Mallory, so he decrypts the message with his private key and then tries to verify Mallory's signature by decrypting it with Mallory's public key. The resultant message, which is pure gibberish, is:

$$E_M[D_B[E_B[D_A[M]]]] = E_M[D_A[M]]$$

Even so, Bob goes on with the protocol and sends Mallory a receipt:

$$E_M(D_B(E_M(D_A(M))))$$

Now, all Mallory has to do is decrypt the message with his private key, encrypt it with Bob's public key, decrypt it again with his private key, and encrypt it with Alice's public key. *Voilà!* Mallory has *M*.

It is not unreasonable to imagine that Bob may automatically send Mallory a receipt. This protocol may be embedded in his communications software, for example, and send receipts automatically. It is this willingness to acknowledge the receipt of gibberish that creates the insecurity. If Bob checked the message for comprehensibility before sending a receipt, he could avoid this security problem.

There are enhancements to this attack that allow Mallory to send Bob a different message from the one he eavesdropped on. Never sign arbitrary messages from other people or decrypt arbitrary messages and give the results to other people.

Foiling the Resend Attack

The attack just described works because the encrypting operation is the same as the signature-verifying operation and the decryption operation is the same as the signature operation. A secure protocol would use even a slightly different operation for encryption and digital signatures. Using different keys for each operation solves the problem, as does using different algorithms for each operation; as do timestamps, which make the incoming message and the outgoing message different; as do digital signatures with one-way hash functions (see Section 2.6).

In general, then, the following protocol is secure as the public-key algorithm used:

(1) Alice signs a message.

(2) Alice encrypts the message and signature with Bob's public key (using a different encryption algorithm than for the signature) and sends it to Bob.

(3) Bob decrypts the message with his private key.

(4) Bob verifies Alice's signature.

Attacks against Public-Key Cryptography

In all these public-key cryptography protocols, I glossed over how Alice gets Bob's public key. Section 3.1 discusses this in detail, but it is worth mentioning here.

The easiest way to get someone's public key is from a secure database somewhere. The database has to be public, so that anyone can get anyone else's public key. The database also has to be protected from write-access by anyone except Trent; otherwise Mallory could substitute any public key for Bob's. After he did that, Bob couldn't read messages addressed to him, but Mallory could.

Even if the public keys are stored in a secure database, Mallory could still substitute one for another during transmission. To prevent this, Trent can sign each public key with his own private key. Trent, when used in this manner, is often known as a **Key Certification Authority** or **Key Distribution Center** (**KDC**). In practical implementations, the KDC signs a compound message consisting of the user's

name, his public key, and any other important information about the user. This signed compound message is stored in the KDC's database. When Alice gets Bob's key, she verifies the KDC's signature to assure herself of the key's validity.

In the final analysis, this is not making things impossible for Mallory, only more difficult. Alice still has the KDC's public key stored somewhere. Mallory would have to substitute his own public key for that key, corrupt the database, and substitute his own keys for the valid keys (all signed with his private key as if he were the KDC), and then he's in business. But, even paper-based signatures can be forged if Mallory goes to enough trouble. Key exchange will be discussed in minute detail in Section 3.1.

2.8 RANDOM AND PSEUDO-RANDOM-SEQUENCE GENERATION

Why even bother with random-number generation in a book on cryptography? There's already a random-number generator built into most every compiler, a mere function call away. Why not use that? Unfortunately, those random-number generators are almost definitely not secure enough for cryptography, and probably not even very random. Most of them are embarrassingly bad.

Random-number generators are not random because they don't have to be. Most simple applications, like computer games, need so few random numbers that they hardly notice. However, cryptography is extremely sensitive to the properties of random-number generators. Use a poor random-number generator and you start getting weird correlations and strange results [1231,1238]. If you are depending on your random-number generator for security, weird correlations and strange results are the last things you want.

The problem is that a random-number generator doesn't produce a random sequence. It probably doesn't produce anything that looks even remotely like a random sequence. Of course, it is impossible to produce something truly random on a computer. Donald Knuth quotes John von Neumann as saying: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin" [863]. Computers are deterministic beasts: Stuff goes in one end, completely predictable operations occur inside, and different stuff comes out the other end. Put the same stuff in on two separate occasions and the same stuff comes out both times. Put the same stuff into two identical computers, and the same stuff comes out of both of them. A computer can only be in a finite number of states (a large finite number, but a finite number nonetheless), and the stuff that comes out will always be a deterministic function of the stuff that went in and the computer's current state. That means that any random-number generator on a computer (at least, on a finite-state machine) is, by definition, periodic. Anything that is periodic is, by definition, predictable. And if something is predictable, it can't be random. A true random-number generator requires some random input; a computer can't provide that.

Pseudo-Random Sequences

The best a computer can produce is a **pseudo-random-sequence generator**. What's that? Many people have taken a stab at defining this formally, but I'll hand-wave here. A pseudo-random sequence is one that looks random. The sequence's period

This is a news release archive from 1996. Substantive information, including information about ABA policy, as well as contact information and links to other Web pages, may not be current. For the most recent ABA news, visit the Division for Media Relations & Communication Services.

Release: Immediate
Contact: Chris Tozer
Email: abanews@abanet.org
URL: http://www.abanet.org/media/home.html

ABA Section Creates First Digital Signature Guidelines To Aid In Security Of The Internet

CHICAGO, Dec. 20 -- People are "surfing" the Internet for knowledge, amusement, and to make small purchases using secure browsers such as Netscape and Microsoft's Internet Explorer , which can encrypt credit card numbers. Very soon, however, the Internet will be used to conduct serious commercial transactions where it is not enough to merely encrypt the message but to make sure that the message originates from the entity allegedly sending it and has not been altered or garbled in transit.

Those engaging in such commerce will want to know that the message and the electronic "signature" attached to the message can both be verified and can be used in court to bind the "signer" to the deal. Looking ahead, the American Bar Association Section of Science and Technology has produced the first legal overview of the use of cryptology, electronic signatures, and entity authentication over an open network like the Internet (called "open" because anyone can "tap" the Internet and intercept electronic traffic flowing over it). The resulting document is called the **Digital Signature Guidelines**, now available from the ABA.

Business owners with an interest in using the Internet for commercial transactions -- which should include virtually every business -- need a basic understanding of the process and the legal principles underlying such commerce. The Guidelines begin with a tutorial that describes in simple terms the technological elements of the public key encryption system.

Public key cryptosystems are based on the use of an associated key pair: a private key that is kept private, and a public key that can be published. The system allows secret messages to be sent over insecure channels like the Internet. By encrypting the messages using the intended recipient's openly available public key, the sender can be certain that only the

intended recipient -- the holder of the associated private key -- will be able to decrypt the message.

The balance of the document describes a system for ensuring the identity of the holder of a private key, for making digital signatures as usable in commerce and in legal proceedings as a written signature on paper, and for ascribing appropriate responsibility to those engaged in electronic commerce should one of the parties involved deny liability under the transaction. Essential to the process is the concept of a Trusted Third-Party. Such parties will investigate to assure themselves, and the public, of the link between the public key and the holder of the private key, authenticate dates and times of transactions, and electronically publish reports of private keys that are no longer reliable. All of this -- the transaction, the signatures, the authentication, the dating, and so on -- is electronic and requires no paper and no warehouses full of documents, and can be accomplished with the speed that is now essential to world-wide commerce.

The Digital Signature Guidelines are available for $34.95, plus shipping and handling, from the ABA Service Center, P.O. Box 10892, Chicago, Ill. 60610-0892, phone: 1-800-285-2221, E-mail: abasvcctr@abanet.org. The product code is 545-0012.

Return to the Media Relations home page | Return to the press release listings

ABA Copyright Statement ABA Privacy Statement Contact the ABA

American Bar Association
Section of Science and Technology
Information Security Committee

Digital Signature Guidelines Tutorial

Digital Signature Guidelines Press Release

Tutorial

In today's commercial environment, establishing a framework for the authentication <1> of computer-based information requires a familiarity with concepts and professional skills from both the legal and computer security fields. Combining these two disciplines is not an easy task. Concepts from the information security field often correspond only loosely to concepts from the legal field, even in situations where the terminology is similar. For example, from the information security point of view, "digital signature" means the result of applying to specific information certain specific technical processes described below. The historical legal concept of "signature" is broader. It recognizes any mark made with the intention of authenticating the marked document. <2> In a digital setting, today's broad legal concept of "signature" may well include markings as diverse as digitized images of paper signatures, typed notations such as "/s/ John Smith," or even addressing notations, such as electronic mail origination headers.

From an information security viewpoint, these simple "electronic signatures" are distinct from the "digital signatures" described in this tutorial and in the technical literature, although "digital signature" is sometimes used to mean any form of computer- based signature. These Guidelines use "digital signature" only as it is used in information security terminology, as meaning the result of applying the technical processes described in this tutorial.

To explain the value of digital signatures in legal applications, this tutorial begins with an overview of the legal significance of signatures. It then sets forth the basics of digital signature technology, and examines how, with some legal and institutional infrastructure, digital signature technology can be applied as a robust computer-based alternative to traditional signatures.

Signatures and the Law

A signature is not part of the substance of a transaction, but rather of its represen tation or form. Signing writings serve the following general purposes:<3>

- **Evidence:** A signature authenticates a writing by identifying the signer with the signed document. When the signer makes a mark in a distinctive manner, the writing becomes attributable to the signer.<4>
- **Ceremony:** The act of signing a document calls to the signer's attention the legal significance of the signer's act, and thereby helps prevent "inconsiderate engagements.<5>
- **Approval:** In certain contexts defined by law or custom, a signature expresses the signer's approval or authorization of the writing, or the signer's intention that it have legal effect.<6>
- **Efficiency and logistics:** A signature on a written document often imparts a sense of clarity and finality to the transaction and may lessen the subsequent need to inquire beyond the face of a document.<7> Negotiable instruments, for example, rely upon formal requirements, including a signature, for their ability to change hands with ease, rapidity, and minimal interruption.<8>

The formal requirements for legal transactions, including the need for signatures, vary in different legal systems, and also vary with the passage of time. There is also variance in the legal consequences of failure to cast the transaction in a required form. The statute of frauds of the common law tradition, for example, does not render a transaction invalid for lack of a "writing signed by the party to be charged," but rather makes it unenforceable in court,<9> a distinction which has caused the practical application of the statute to be greatly limited in case law.

During this century, most legal systems have reduced formal requirements,<10> or at least have minimized the consequences of failure to satisfy formal requirements. Nevertheless, sound practice still calls for transactions to be formalized in a manner which assures the parties of their validity and enforceability.<11> In current practice, formalization usually involves documenting the transaction on paper and signing or authenticating the paper. Traditional methods, however, are undergoing fundamental change. Documents continue to be written on paper, but sometimes

merely to satisfy the need for a legally recognized form. In many instances, the information exchanged to effect a transaction never takes paper form. Computer-based information can also be utilized differently than its paper counterpart. For example, computers can "read" digital information and transform the information or take programmable actions based on the information. Information stored as bits rather than as atoms of ink and paper can travel near the speed of light, may be duplicated without limit and with insignificant cost.

Although the basic nature of transactions has not changed, the law has only begun to adapt to advances in technology. The legal and business communities must develop rules and practices which use new technology to achieve and surpass the effects historically expected from paper forms.

To achieve the basic purposes of signatures outlined above, a signature must have the following attributes:<12>

- **Signer authentication:** A signature should indicate who signed a document, message or record,<13> and should be difficult for another person to produce without authorization.
- **Document authentication:** <14> A signature should identify what is signed, <15> making it impracticable to falsify or alter either the signed matter or the signature without detection.

Signer authentication and document authentication are tools used to exclude impersonators and forgers and are essential ingredients of what is often called a "nonrepudiation service" in the terminology of the information security profession. A nonrepudiation service provides assurance of the origin or delivery of data in order to protect the sender against false denial by the recipient that the data has been received, or to protect the recipient against false denial by the sender that the data has been sent. <16> Thus, a nonrepudiation service provides evidence to prevent a person from unilaterally modifying or terminating legal obligations arising out of a transaction effected by computer-based means. <17>

- **Affirmative act:** The affixing of the signature should be an affirmative act which serves the ceremonial and approval functions of a signature and establishes the sense of having legally consummated a transaction.
- **Efficiency:** Optimally, a signature and its creation and verification processes should provide the greatest possible assurance of both signer authenticity and document authenticy, with the least possible expenditure of resources.

Digital signature technology generally surpasses paper technology in all these attributes. <18> To understand why, one must first understand how digital

signature technology works.

How Digital Signature Technology Works

Digital signatures are created and verified by cryptography, the branch of applied mathematics that concerns itself with transforming messages into seemingly unintelligible forms and back again. Digital signatures use what is known as "public key cryptography," which employs an algorithm using two different but mathematically related "keys;" one for creating a digital signature or transforming data into a seemingly unintelligible form, and another key for verifying a digital signature or returning the message to its original form. <19> Computer equipment and software utilizing two such keys are often collectively termed an "asymmetric cryptosystem."

The complementary keys of an asymmetric cryptosystem for digital signatures are arbitrarily termed the private key, which is known only to the signer <20> and used to create the digital signature, and the public key, which is ordinarily more widely known and is used by a relying party to verify the digital signature. If many people need to verify the signer's digital signatures, the public key must be available or distributed to all of them, perhaps by publication in an on-line repository or directory where it is easily accessible. Although the keys <21> of the pair are mathematically related, if the asymmetric cryptosystem has been designed and implemented securely <22> it is "computationally infeasible <23> to derive the private key from knowledge of the public key. Thus, although many people may know the public key of a given signer and use it to verify that signer's signatures, they cannot discover that signer's private key and use it to forge digital signatures. This is sometimes referred to as the principle of "irreversibility."

Another fundamental process, termed a "hash function," is used in both creating and verifying a digital signature. A hash function is an algorithm which creates a digital representation or "fingerprint" in the form of a "hash value" or "hash result" of a standard length which is usually much smaller than the message but nevertheless substantially unique to it. <24> Any change to the message invariably produces a different hash result when the same hash function is used. In the case of a secure hash function, sometimes termed a "one-way hash function," it is computationally infeasible <25> to derive the original message from knowledge of its hash value. Hash functions therefore enable the software for creating digital signatures to operate on smaller and predictable amounts of data, while still providing robust evidentiary correlation to the original message content, thereby efficiently providing assurance that there has been no modification of the message since it was digitally signed.

Thus, use of digital signatures usually involves two processes, one performed by the signer and the other by the receiver of the digital signature:

- **Digital signature creation** uses a hash result derived from and unique to both the signed message and a given private key. For the hash result to be secure, there must be only a negligible possibility that the same digital signature could be created by the combination of any other message or private key.
- **Digital signature verification** is the process of checking the digital signature by reference to the original message and a given public key, thereby determining whether the digital signa ture was created for that same message using the private key that corresponds to the referenced public key.

To sign a document or any other item of information, the signer first delimits precisely the borders of what is to be signed. The delimited information to be signed is termed the "message" in these Guidelines. Then a hash function in the signer's software computes a hash result unique (for all practical purposes) to the message. The signer's software then transforms the hash result into a digital signature using the signer's private key. <26> The resulting digital signature is thus unique to both the message and the private key used to create it.

Typically, a digital signature (a digitally signed hash result of the message) is attached to its message and stored or transmitted with its message. However, it may also be sent or stored as a separate data element, so long as it maintains a reliable association with its message. Since a digital signature is unique to its message, it is useless if wholly disassociated from its message.

Verification of a digital signature is accomplished by computing a new hash result of the original message by means of the same hash function used to create the digital signature. Then, using the public key and the new hash result, the verifier checks: (1) whether the digital signature was created using the corresponding private key; and (2) whether the newly computed hash result matches the original hash result which was transformed into the digital signature during the signing process. The verification software will confirm the digital signature as "verified" if: (1) the signer's private key was used to digitally sign the message, which is known to be the case if the signer's public key was used to verify the signature because the signer's public key will verify only a digital signature created with the signer's private key; <27> and (2) the message was unaltered, which is known to be the case if the hash result computed by the verifier is identical to the hash result extracted from the digital signature during the verification process.

Various asymmetric cryptosystems create and verify digital signatures using different algorithms and procedures, but share this overall operational pattern.

The processes of creating a digital signature and verifying it accomplish the essential effects desired of a signature for many legal purposes:

- **Signer authentication:** If a public and private key pair is associated with an identified signer, the digital signature attributes the message to the signer. The digital signature cannot be forged, unless the signer loses control of the private key (a "compromise" of the private key), such as by divulging it or losing the media or device in which it is contained.
- Message authentication: The digital signature also identifies the signed message, typically with far greater certainty and precision than paper signatures. Verification reveals any tampering, since the comparison of the hash results (one made at signing and the other made at verifying) shows whether the message is the same as when signed.
- **Affirmative act:** Creating a digital signature requires the signer to use the signer's private key. This act can perform the "ceremonial" function of alerting the signer to the fact that the signer is consummating a transaction with legal consequences. <28>
- **Efficiency:** The processes of creating and verifying a digital signature provide a high level of assurance that the digital signature is genuinely the signer's. As with the case of modern electronic data interchange ("EDI") the creation and verification processes are capable of complete automation (sometimes referred to as "machinable"), with human interaction required on an exception basis only. Compared to paper methods such as checking specimen signature cards -- methods so tedious and labor-intensive that they are rarely actually used in practice -- digital signatures yield a high degree of assurance without adding greatly to the resources required for processing.

The processes used for digital signatures have undergone thorough technological peer review for over a decade. Digital signatures have been accepted in several national and international standards developed in cooperation with and accepted by many corporations, banks, and government agencies. <29> The likelihood of malfunction or a security problem in a digital signature cryptosystem designed and implemented as prescribed in the industry standards is extremely remote, <30> and is far less than the risk of undetected forgery or alteration on paper or of using other less secure electronic signature techniques.

Public Key Certificates

To verify a digital signature, the verifier must have access to the signer's public key and have assurance that it corresponds to the signer's private key. However, a public and private key pair has no intrinsic association with any person; it is simply

a pair of numbers. Some convincing strategy is necessary to reliably associate a particular person or entity to the key pair.

In a transaction involving only two parties, each party can simply communicate (by a relatively secure "out-of-band" channel such as a courier or a secure voice telephone) the public key of the key pair each party will use. Such an identification strategy is no small task, especially when the parties are geographically distant from each other, normally conduct communication over a convenient but insecure channel such as the Internet, are not natural persons but rather corporations or similar artificial entities, and act through agents whose authority must be ascertained. As electronic commerce increasingly moves from a bilateral setting to the many-on-many architecture of the World Wide Web on the Internet, where significant transactions will occur among strangers who have no prior contractual relationship and will never deal with each other again, the problem of authentication/nonrepudiation becomes not merely one of efficiency, but also of reliability. An open system of communication such as the Internet needs a system of identity authentication to handle this scenario.

To that end, a prospective signer might issue a public statement, such as: "Signatures verifiable by the following public key are mine." However, others doing business with the signer may for good reason be unwilling to accept the statement, especially where there is no prior contract establishing the legal effect of that published statement with certainty. A party relying upon such an unsupported published statement in an open system would run a great risk of trusting a phantom or an imposter, or of attempting to disprove a false denial of a digital signature ("nonrepudiation") if a transaction should turn out to prove disadvantageous for the purported signer.

The solution to these problems is the use of one or more trusted third parties to associate an identified signer with a specific public key. <31> That trusted third party is referred to as a "certification authority" in most technical standards and in these Guidelines.

To associate a key pair with a prospective signer, a certification authority issues a certificate, an electronic record which lists a public key as the "subject" of the certificate, and confirms that the prospective signer identified in the certificate holds the corresponding private key. The prospective signer is termed the "subscriber. <32> A certificate's principal function is to bind a key pair with a particular subscriber. A "recipient" of the certificate desiring to rely upon a digital signature created by the subscriber named in the certificate (whereupon the recipient becomes a "relying party") can use the public key listed in the certificate to verify that the digital signature was created with the corresponding

corresponding private key. <33> If such verification is successful, this chain of reasoning provides assurance that the corresponding private key is held by the subscriber named in the certificate, and that the digital signature was created by that particular subscriber.

To assure both message and identity authenticity of the certificate, the certification authority digitally signs it. The issuing certification authority's digital signature on the certificate can be verified by using the public key of the certification authority listed in another certificate by another certificate authority (which may but need not be on a higher level in a hierarchy) <34>, and that other certificate can in turn be authenticated by the public key listed in yet another certificate, and so on, until the person relying on the digital signature is adequately assured of its genuineness. In each case, the issuing certification authority must digitally sign its own certificate during the operational period of the other certificate used to verify the certification authority's digital signature.

A digital signature, whether created by a subscriber to authenticate a message or by a certification authority to authenticate its certificate (in effect a specialized message) should be reliably time-stamped to allow the verifier to determine reliably whether the digital signature was created during the "operational period" stated in the certificate, which is a condition upon verifiability of a digital signature under these Guidelines. <35>

To make a public key and its identification with a specific subscriber readily available for use in verification, the certificate may be published in a repository or made available by other means. Repositories are on-line databases of certificates and other information available for retrieval and use in verifying digital signatures. Retrieval can be accomplished automatically by having the verification program directly inquire of the repository to obtain certificates as needed.

Once issued, a certificate may prove to be unreliable, such as in situations where the subscriber misrepresents his identity to the certification authority. In other situations, a certificate may be reliable enough when issued but come to be unreliable sometime thereafter. If the subscriber loses control of the private key ("compromise" of the private key), the certificate has become unreliable, and the certification authority (either with or without the subscriber's request depending on the circumstances) may suspend (temporarily invalidate) or revoke (permanently invalidate) the certificate. Immediately upon suspending or revoking a certificate, the certification authority must publish notice of the revocation or suspension or notify persons who inquire or who are known to have received a digital signature verifiable by reference to the unreliable certificate.

Challenges and Opportunities

The prospect of fully implementing digital signatures in general commerce presents both benefits and costs. The costs consist mainly of:

- **Institutional overhead:** The cost of establishing and utilizing certification authorities, repositories, and other important services, as well as assuring quality in the performance of their functions.
- **Subscriber and Relying Party Costs:** A digital signer will require software, and will probably have to pay a certification authority some price to issue a certificate. Hardware to secure the subscriber's private key may also be advisable. Persons relying on digital signatures will incur expenses for verification software and perhaps for access to certificates and certificate revocation lists (CRL) in a repository.

On the plus side, the principal advantage to be gained is more reliable authentication of messages. Digital signatures, if properly implemented and utilized offer promising solutions to the problems of:

- **Imposters**, by minimizing the risk of dealing with imposters or persons who attempt to escape responsibility by claiming to have been impersonated;
- **Message integrity**, by minimizing the risk of undetected message tampering and forgery, and of false claims that a message was altered after it was sent;
- **Formal legal requirements**, by strengthening the view that legal requirements of form, such as writing, signature, and an original document, are satisfied, since digital signatures are functionally on a par with, or superior to paper forms; and
- **Open systems**, by retaining a high degree of information security, even for information sent over open, insecure, but inexpensive and widely used channels.

Digital Signature Guidelines

Tutorial Footnotes

<1> 1For purposes of these Guidelines, authentication is generally the process used to confirm the identity of a person or to prove the integrity of specific information. More specifically, in the case of a message, authentication involves determining its source and providing assurance that the message has not been modified or replaced in transit. *See* Guideline 28 (authentication).

<2> 2*See, e.g.*, U. C. C. § 1-201(39) (1992).

<3> 3This list is not exhaustive. For example, Restatement (Second) of Contracts notes another function, termed the "deterrent function," which seeks to "discourage transactions of doubtful utility." Restatement (Second) of Contracts § 72 comment c (1981). Professor Perillo notes earmarking of intent, clarification, managerial efficiency, publicity, education, as well as taxation and regulation as functions served by the statute of frauds. Joseph M. Perillo, *The Statute of Frauds in the Light of the Functions and Dysfunctions of Form*, 43 Fordham L. Rev. 39, 48-64 (1974) (hereinafter "Perillo").

<4> 4*See* Restatement (Second) of Contracts, statutory note preceding § 110 (1982) (summarizing purpose of the statute of frauds, which includes a signature requirement); Lon L. Fuller, *Consideration and Form*, 41 Colum. L. Rev. 799, 800 (1941) (hereinafter "Fuller"); 6 Jeremy Bentham, The Works of Jeremy Bentham 508-85 (Bowring ed. 1962) (1839) (Bentham called forms serving evidentiary functions "preappointed [*i.e.*, made in advance] evidence"). A handwritten signature creates probative evidence in part because of the chemical properties of ink that make it adhere to paper, and because handwriting style is quite unique to the signer. Perillo, *supra* note 3, at 64-69. *See* U. C. C. § 1-201(39) ("?Signed' includes any symbol executed or adopted by a party with present intention to authenticate a writing.").

<5>" 52 John Austin, Lectures on Jurisprudence 939-44 (4th ed. 1873); Restatement (Second) of Contracts § 72 comment c (1982) and statutory note preceding § 110 (1982) (what is here termed a "ceremonial" function is termed a "cautionary" function in the Restatement); Perillo, *supra* note 3, at 53-56; Fuller, *supra* note 4, at 800; Rudolf von Jhering, Geist des römischen Rechts § 45, at 494-98 (8th ed. 1883) (hereinafter "Jhering").

<6> 6*See Model Law on Electronic Commerce,* United Nations Commission on

International Trade Law (UNCITRAL), 29th Sess., art. 7(1), at 3, U.N. Doc. A/CN.9/XXIX/CRP.1/Add.13 (1996) ("Where a law requires a signature of a person, that requirement is met in relation to a data message if: (a) a method is used to identify that person and to indicate that person's approval of the information contained in the data message...."); *Draft Model Law on Legal Aspects of Electronic Data Interchange (EDI) and Related Means of Data Communication,* United Nations Commission on International Trade Law (UNCITRAL), 28th Sess., art. 6, at 44, U.N. Doc. A/CN.9/406 (1994). For example, a signature on a written contract customarily indicates the signer's assent. A signature on the back of a check is customarily taken as an endorsement. *See* U.C.C. § 3-204 (1990).

<7> 7*See* Perillo, *supra* note 3, at 50-53; Fuller, *supra* note 4, at 801-802; Jhering, *supra* note 5, at 494-97 (analogizing the form of a legal transaction to minting of coins, which serves to make their metal content and weight apparent without further examination). The notion of clarity and finality provided by a form are largely predicated on the fact that the form provides good evidence. The basic premise of the efficiency and logistical function is that a signed, written document is such a good indicator of what the transaction is, that the transaction should be considered to be as the signed document says. The moment of signing the document thus becomes decisive.

<8> 8*See, e.g.*, U.C.C. § 3-401 (1990) (a person is not liable on an instrument unless the person signed it); *see generally* U.C.C. § 3-104 (1990) (requirements for negotiability).

<9> 92 Arthur L. Corbin, Corbin on Contracts § 279, at 20-23 (1950). In English law, the original 1677 statute of frauds was repealed in 1954 by the Law Reform (Enforcement of Contracts) Act, 2 & 3 Eliz. II, c. 34, except for its suretyship and real property provisions. However, it remains in force throughout the United States and in much of the British Commonwealth outside the United Kingdom.

<10> 10*See* Perillo *supra* note 3, at 41-42. In Anglo-American law, there are many examples of the trend away from formal requirements. For example, the common law seal has little remaining significance. *See* Restatement (Second) of Contracts, statutory note preceding § 95 (1982). Case law has greatly limited the effects of the statute of frauds through the part performance doctrine, promissory and equitable estoppel (*e.g.* Monarco v. Lo Greco, 35 Cal. 2d 621, 220 P.2d 737 (1950) (Traynor, J.)), leniency in determining what constitutes a sufficient memorandum, and by permitting restitution and reformation of a contract within the statute of frauds. For a classic examination of the advantages and disadvantages of formal requirements, *see* Jhering, *supra* note 5, at 470-504.

<11> 11Michael Braunstein, *Remedy, Reason, and the Statute of Frauds: A Critical Economic Analysis,* 1989 Utah L. Rev. 383, 423-26 (1989); Jhering, *supra*, note 5, at 474 (inattention to legally appropriate form for expressing intent exacts its own consequences

("rächt sich selber")).

<12> 12*Cf.* The U.S. Comptroller General's rationale for accepting digital signatures as sufficient for government contracts under 31 U.S.C. 1501(a)(1): "The electronic symbol proposed for use by certifying officers . . . embodied all of the attributes of a valid, acceptable signature: it was unique to the certifying officer, capable of verification, and under his sole control such that one might presume from its use that the certifying officer, just as if he had written his name in his own hand, intended to be bound." *In re National Institute of Standards and Technology — Use of Electronic Data Interchange to Create Valid Obligations,* file B-245714 (Comptroller Gen'l, 1991).

<13> 13In U.C.C. Art. 2B (May 3, 1996 Draft), "Record" is defined by § 2B-102(30) as "information that is inscribed on a tangible medium or that is stored in an electronic or other medium and is retrievable in perceivable form.." *See also, Model Law on Electronic Commerce,* United Nations Commission on International Trade Law (UNCITRAL), 29th Sess., art. 7(1), at 3, U.N. Doc. A/CN.9/XXIX/CRP.1/Add.13 (1996) ("Where a law requires a signature of a person, that requirement is met in relation to a data message if: (a) a method is used to identify that person and to indicate that person's approval of the information contained in the data message...."). Throughout these Guidelines "message" means the digital representation of information (generally, computer-based information), "document" means information inscribed on a tangible medium (generally paper-based information), and "record" can be used to refer to a message or to a document, consistent with the definition of "record" in U.C.C. § 2B-102(30), *supra*, this footnote.

<14> 14Document authentication is similar to the security service of message integrity which provides assurance that the information signed has not been altered. *See* Guideline 35 (authentication).

<15> 15A paper signature identifies the signed matter less than perfectly. Ordinarily, the signature appears below what is signed, and the physical dimensions of the paper and the regular layout of the text are relied upon to indicate alteration. However, those mechanisms are not enough to prevent difficult factual questions from arising. *See, e.g.*, Citizens Nat'l Bank of Downers Grove v. Morman, 78 Ill. App. 3d 1037, 398 N.E.2d 49 (1979); Newell v. Edwards, 7 N.C. App. 650, 173 S.E.2d 504 (1970); Zions First Nat'l Bank v. Rocky Mountain Irrigation, Inc., 795 P.2d 658, 660-63 (Utah 1990); Lembo v. Federici, 62 Wash. 2d 972, 385 P.2d 312 (1963).

<16> 16*Information Technology - Security Frameworks in Open Systems - Non-Repudiation Framework* (also *ITU-T Recommendation X.813*), ISO/IEC 10181-4 (1996); Warwick Ford, Computer Communications Security: Principles, Standard Protocols & Techniques 29-30 (1994) (1994) (hereinafter "Ford"); Michael S. Baum, Federal Certification Authority Liability and Policy: Law and Policy of Certificate-Based

Public Key and Digital Signatures 9 (National Institute of Standards and Technology 1994) (hereinafter "Baum"). Sender and recipient have a mutual incentive to use an authentication service to exclude disruption from third party intrusion, but a nonrepudiation service is used by sender or recipient adversely against the other, when one wishes to deny having made or received a communication and the other has an incentive to prove that it was made or received. *See* Charles R. Merrill, *A Cryptographic Primer*, The Internet and Business: A Lawyer's Guide to the Emerging Legal Issues 14 (Joseph F. Ruh, Jr., ed., The Computer Law Association 1996).

<17> 17A nonrepudiation service provides only proof of facts to defend against an opponent's effort to avoid a transaction. *See* Baum, *supra* note 16, at 6 (1994). *See* Guideline 1.20 (nonrepudiation), particularly Comment 1.20.1.

<18> 18For a more thorough examination of properties desirable in a digital signature, *see generally* Bruce Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C §2.6, 33-40 (2d ed. 1996) (hereinafter "Schneier"); Mitchell, Piper & Wild, Digital Signatures, in Contemporary Cryptology: The Science of Information Integrity 325, 341-46 (Gustavas Simmons ed., 1991).

<19> 19In contrast with public key cryptography, "conventional," "single key," or "symmetric cryptography" uses the same single key to "encrypt" "plaintext" into "ciphertext," and to "decrypt" it from ciphertext back to plaintext.

<20> 20Of course, the holder of the private key may choose to divulge it, or may lose control of it (often called "compromise"), and thereby make forgery possible. The Guidelines seek to address this problem in two ways, (1) by requiring a subscriber, who holds the private key, to use a degree of care in its safekeeping, and (2) enabling the subscriber to disassociate himself from the key by temporarily suspending or permanently revoking his certificate and publishing these actions in a "certificate revocation list," or "CRL". A variety of methods are available for securing the private key. The safer methods store the private key in a "cryptographic token" (one example is a "smart card") which executes the signature program within an internal microprocessing chip, so that the private key is never divulged outside the token and does not pass into the main memory or processor of the signer's computer. The signer must typically present to the token some authenticating information, such as a password, pass phrase, or personal identification number, for the token to run a process requiring access to the private key. In addition, this token must be physically produced, and biometric authentication such as fingerprints or retinal scan can assure the physical presence of the token's authorized holder. There are also software-based schemes for protecting the security of the private key, generally less secure than hardware schemes, but providing adequate security for many types of applications. *See generally* Schneier, *supra* note 18, at § 2.7, 41-44.

<21> 21Many cryptographic systems will function securely only if the keys are lengthy and complex, too lengthy and complex for a person to easily remember or use.

<22> 22*See generally* Ford, *supra* note 16, at 71-75; Charlie Kaufman, Radia Perlman & Mike Speciner, Network Security: Private Communication in a Public World 48-56 (1995) (hereinafter "Kaufman, et al., Network Security").

<23>" 23"Computationally infeasible" is a relative concept based on the value of the data protected, the computing overhead required to protect it, the length of time it needs to be protected, and the cost and time required to attack the data, with such factors assessed both currently and in the light of future technological advance. *See generally,* Schneier, *supra* note 18, at § 7.5, 166-67.

<24> 24*See generally* Ford, *supra* note 16, at 75-84. Computer Communications Security 75-84 (1994); Kaufman, et al., Network Security, *supra* note 22, at 101-27; Nechvatal, *Public Key Cryptography, in* Comtemporary Crypt ology: The Science of Information Integrity 179, 199-202 (Gustavas Simmons ed. 1991); Schneier, *supra* note 18, §§ 18.1-18.14, 429-459.

<25> 25 "Because hash functions are typically many-to-one, we cannot use them to determine with certainty that the two [input] strings are equal, but we can use them to get a reasonable assurance of accuracy." Schneier, *supra* note 18, at § 2.4, 30-31. It is extremely improbable that two messages will produce the same hash result. *See* Kaufman, et al., Network Security, *supra* note 22, at 102.

<26> 26This transformation is sometimes described as "encryption," which is inaccurate terminology, because the message itself does not need to be confidential. Confidentiality can be provided as an optional feature of digital signature technologies, but the separate and distinct security service of confidentiality is not central to the security services of signer authentication and document authentication, and is thus outside the scope and focus of these guidelines.

<27> 27Because of the mathematical relationship between the public and private keys which correspond to each other as a key pair. Schneier, *supra* note 18, at § 2.6, 34-41.

<28> 28If the person "signing" the message is not a human being but a device under the control of a human being as permitted by these Guidelines, this ceremonial function may be undermined.

<29> 29As of this writing, the following jurisdictions have enacted or introduced some form of legislation dealing with digital signatures or electronic signatures:

1996 Arizona House Bill 2444, amending Ariz. Rev. Stat. Ann. § 41-121 (1996) (enacted)

(URL:http://www.azleg.state.az.us/legtext/42leg/2r/laws/0213.htm);

Cal. Gov't Code § 16.5 (West 1995) (enacted) (URL:http://www.sen.ca.gov);

Conn. Gen. Stat. § 19a-25a (1994) (enacted);

1996 Fla. Senate Bill 942 (enacted)
(URL:http://www.scri.fsu.edu/fla-leg/senate-1996/sb0942er.html);

1995 Ga. Senate Resolution 621 and House Resolution 1256 (pending); 1995 Ga. Senate Bill 736 (died in committee);

1995 Haw. Senate Bill 2401(pending);

1995 Ill. House Bill 3394 (pending);

Iowa Code § 48A.13 (1995) (enacted)
(URL:http://www2.legis.state.ia.us/cgi-bin/IACODE/Code1995SUPPLEMENT.P1;

La. Rev. Stat. Ann § 40:2144 (West 1995) (enacted);

Mich. Senate Bill 939 (pending)
(URL:http://www.coast.net/~misenate/dem/agenda/sig/sb939.html);

1995 N.Y. Senate Bill 7420 (pending) (URL:gopher.senate.state.ny.us);

1996 R.I. House Bill 8125 (pending);

Utah Code Ann. § 46:3 (1996) (URL:http://www.state.ut.us/ccjj/digsig/dsut-act.htm)
("Utah Digital Signature Act");

1996 Va. House Joint Resolution 129 (pending)
(URL:http://www.state.va.us/dlas/ses19961/ful/hj129.htm);

1996 Wash. Senate Bill 6423 (URL:http://leginfo.leg.wa.gov/pub/billinfo/senate/);

1996 Wyo. Senate File12
(URL:gopher://merlin.state.wy.us:70/00/wgov/lb/lb/session/BILLS/1995/Enrolled/Senate_F

Massachusetts is considering digital signature legislation. Telephone interview with Daniel

Greenwood, Esq., Deputy General Counsel, Information Technology Division, Commonwealth of Massachusetts (July 19, 1996).

Germany and Chile are both considering digital signature legislation. *See generally*, on-line public discussion in E-Mail and Electronic Commerce Forum of Lexis Counsel Connect (Jan.-Mar. 1996).

<30> 30Although generally beyond the scope of these Guidelines, we note that current U.S. export restrictions, Department of State, "International Traffic in Arms Regulations (ITAR)," Office of Munitions Control, 22 C.F.R. §§ 120-130 (Nov. 1989), on software which possesses both confidentiality encryption and digital signature capability (or which can be converted into confidentiality encryption software) has caused software providers to intentionally emasculate ("dumb down") algorithms in some of their domestic as well as international products. This is considered by some to have cast doubt upon the "computational infeasibility" assumed by the standards, for digital signature as well as confidentiality encryption software. *See generally*, Schneier, *supra* note 18, at § 25.14, 610-16.

<31> See Schneier, *supra* note 18, at § 8.12, 185-7; Baum, *supra* note 16, at 10-11; *See generally*, A. Michael Froomkin, *The Essential Role of Trusted Third Parties in Electronic Commerce*, 75 Or. L. Rev. 49 (1996).

<32>" 32The subscriber is sometimes also called an "applicant" after applying to a certification authority for a certificate, but before the certificate issuance procedure is completed.

<33> 33The statement in the certificate of the beginning and ending date of the operational period of the certificate also allows a determination of whether or not a time-dated digital signature was created during the operational period of the certificate. A search of the certificate revocation list (CRL) also enables the verifier to determine the certificate has been revoked or suspended earlier than the end of the stated operational period of the certificate. *See* Guidelines 1.22 (operational period) and 1.37 (verify a digital signature).

<34> 34A number of models exist which implement different strategies for the certification of the public keys of certification authorities who issue certificates (sometimes referred to generically as "issuing authorities"). Some examples include (i) a multi-level hierarchical structure back to a single "root," where public keys of issuing authorities are certified by the next higher-level certification authority; (ii) a flatter hierarchical structure where a single "root" might directly certify the public keys of all issuing authorities below it; (iii) a single level of issuing authorities which "cross-certify" each others' public keys; or (iv) a "system in which each issuing authority's public key is certified in some reliable manner without reference to a second certification authority. In a hierarchical system, the public

key of the "root" certification authority is, by definition, self-authenticating.

<35> 35A reliable time-stamp on the certificate also allows a determination as to whether it was created before or after the filing of a revocation or suspension of a certificate in a repository, which not only protects the subscriber who promptly revokes or suspends, but also provides increased assurance of nonrepudiatability by making it more difficult for a fraudulent subscriber to create a certificate and retroactively revoke it after reliance upon the certificate has occurred.